UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO
CURSOS DE CIÊNCIA E ENGENHARIA DE COMPUTAÇÃO
Disciplina de Programação Orientada a Objetos - 2022/2

RELATÓRIO PROJETO FINAL

STRATEGY GAME

por João Vitor Laimer e Vinícius Hallmann
Professor: Felipe de Souza Marques

Introdução

Esse relatório tem como objetivo apresentar o trabalho final da disciplina de Programação Orientada a Objetos desenvolvido por João Vitor Laimer e Vinícius Hallmann. O trabalho consiste na criação de um jogo, chamado de "STRATEGY GAME" desenvolvido em JAVA aplicando os conceitos de Programação Orientada a Objetos.

Descrição do jogo

"STRATEGY GAME" é um jogo de batalha por turnos entre um jogador e o computador em que cada player possui um exército, que são diferenciados pelas cores azul e vermelho, distribuídos no seu lado do campo de batalha num tabuleiro de 5x5. O objetivo do jogo é capturar a bandeira inimiga do outro lado do campo de batalha.

Descrição das funcionalidades implementadas

Durante a criação do projeto vários métodos e funcionalidades foram criadas a fim de ajudar no desenvolvimento do jogo. Podemos dizer que os principais tópicos e que podem ser considerados a base do projeto são:

1. Criação do tabuleiro;
2. Criação das peças;
3. Posicionamento das peças sobre o tabuleiro;
4. Validação de movimento;
5. Combate;

Criação do tabuleiro e criação das peças:

Alguns dos principais pontos sobre a criação do tabuleiro são: Classe célula, célula Factory e o método constroiTabuleiro()

- A classe Célula é responsável por representar cada célula do tabuleiro onde é armazenado seus atributos principais (Peça, Equipe, Coordenadas);
- A Factory de células é responsável por receber como parâmetro qual tipo de célula deve ser criada e a retorna;
- E por último o método constroiTabuleiro() tem como função criar a estrutura básica do jogo a partir da célula factory, criando um tabuleiro 5x5 com células vazias.

Posicionamento das peças sobre o tabuleiro:

O método usado para o posicionamento das peças sobre o tabuleiro é o colocaPeçaNoTabuleiro (Célula botãoClicado, coordenada x e y);

Esse método é responsável pela criação da peça desejada, depois pela atribuição da nova coordenada dessa peça e remoção da antiga célula na mesma coordenada.

Validação de movimento:

Os métodos usados para o movimento das peças são:
validaMovimento(Célula botãoClicado, coordenada x e y), verifica (Celula botão), calculoDistancia(Celula botãoClicado).

- O método verifica() recebe a célula que foi clicada e verifica se aquela célula não é do tipo vazio, caso não seja, retorna true.
- O método calculaDistancia é responsável por receber a célula no tabuleiro que foi clicada e calcular a distância dos pontos entre as coordenadas do botão que o jogador quer mover com a célula onde o ele pretende posicionar. Como é dito nas regras, as peças só podem mover uma casa adjacente na horizontal ou vertical, e para isso é usado a seguinte formula |coordX1 - CoordX2| e |coordY1 - CoordY2| e para o movimento ser válido um desses cálculos precisa dar 1 e outro 0. A única peça que pode burlar essa regra é a peça do tipo Soldado que pode correr pelo tabuleiro, então a verificação de seu movimento é calculado através da verificação de algum obstáculo (aliados, inimigos ou lago) do eixo na qual ele deseja "correr".
- O método que valida o movimento recebe a célula na qual se deseja fazer o movimento e chama o calcula Distância() para saber se o movimento está dentro do valor definido pela regra. Depois ele verifica se célula a clicada é do tipo de peças que não podem se mover (Bandeira, Bomba e Vazio) caso todas verificações estejam corretas, a função retorna true validando o movimento.

Combate

O método combate recebe o tipo da peça selecionada do jogador e a peça selecionada para lutar contra e faz uma verificação sobre as peças que foram jogadas na rodada e retorna qual foi o resultado da luta entre elas

Dificuldades enfrentadas durante o processo da criação do jogo

1. Dificuldade na projeção do trabalho:

   No primeiro momento nós tivemos um pouco de dificuldade em como começar a estruturar o jogo. Na primeira versão do projeto, nós começamos a desenvolver o jogo de uma maneira que a longo prazo para nós não seria eficiente e nem mais fácil, então por isso tivemos que dar um passo para trás, pensar uma outra maneira de como recomeçar com o progresso que nós já tínhamos. A maneira que nós começamos a

desenvolver a de primeiro momento foi usando o Form Design do NetBeans, que no início parecia uma ideia boa que iria facilitar nosso trabalho e desenvolvimento, mas percebemos que perderíamos muito na eficiência e também teríamos dificuldade em escrever um código conciso e que aplicasse os conceitos de programação orientada a objetos.

2. Dificuldade na movimentação das peças:

   Um desafio que nós tivemos foi no desenvolvimento do código em relação ao movimento das peças. Por ser um elemento crucial para o jogo, nós estávamos presos a resolver esse problema e por isso nós ficamos muito tempo amarrados nessa parte do projeto, foram vários códigos escritos até chegarmos em um código que satisfazia com o que queríamos. Após feito essa parte, conseguimos fazer uma delegação de tarefas sobre o desenvolvimento de outras funcionalidades do código e isso levou a uma rápida progressão sobre o resto do desenvolvimento do projeto.

Passos para compilar e rodar o programa gerado

O programa está disponível de duas maneiras

1. Arquivo JAR com o executável do jogo disponível neste link: Arquivo JAR STRATEGY_GAME.jar
   Para executar o programa primeiro é preciso extrair a pasta dist, entrar nela e executar o arquivo MainStrategyGame.jar;

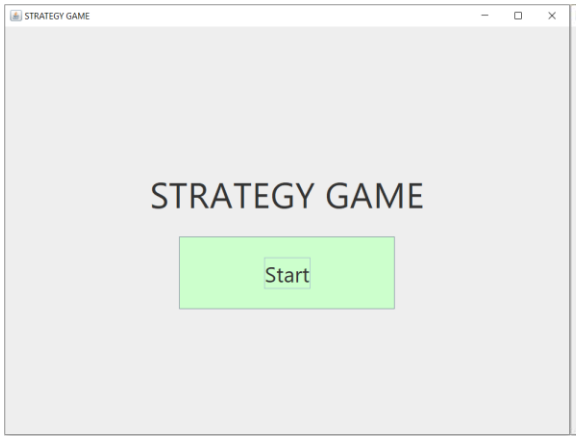
2. Pasta com todos os arquivos do jogo para ser executado no NetBeans: Arquivo MainStrategyGame.jar
   Para executar o programa primeiro é preciso extrair a pasta MainStrategyGame, depois entrar na IDE NetBeans, abrir um novo projeto, selecionar a pasta extraída e depois rodar o jogo pelo Run Project (Tecla F6).

Diagrama de classes

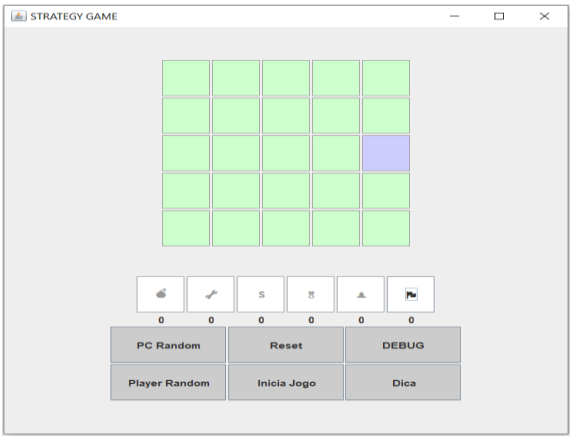
O diagrama de classes pode ser acessado por esse link, para uma melhor visualização aconselhasse abrir o diagrama pelo diagrams.net

Exemplos de utilização

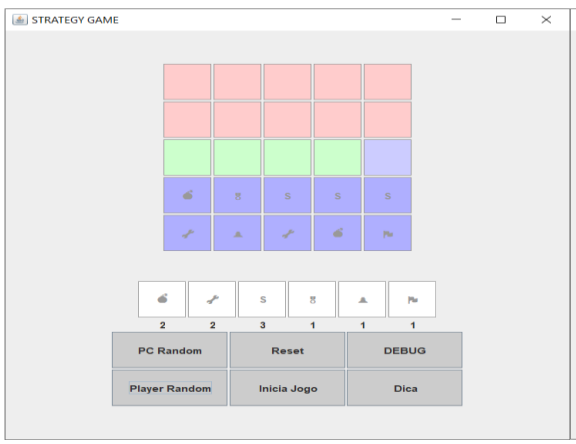
Telas iniciais do jogo

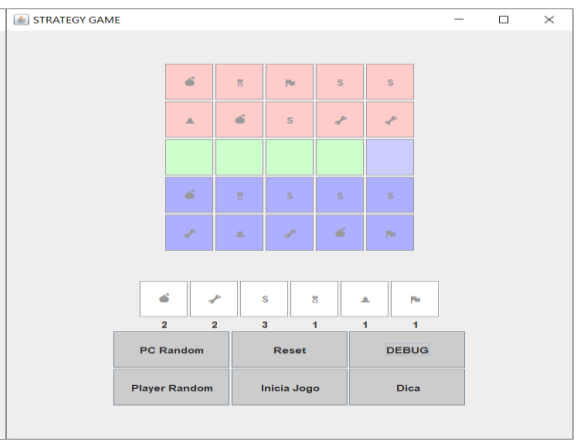


Tabuleiro vazio

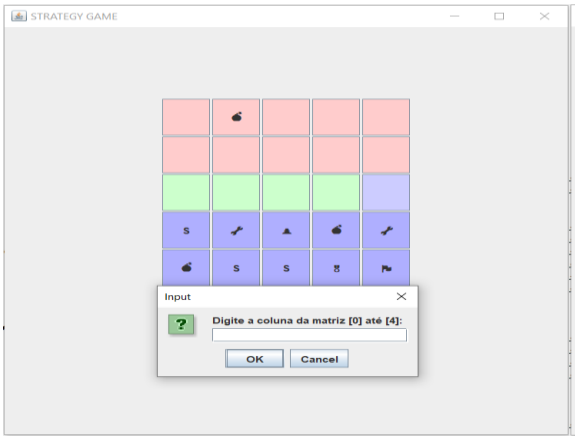


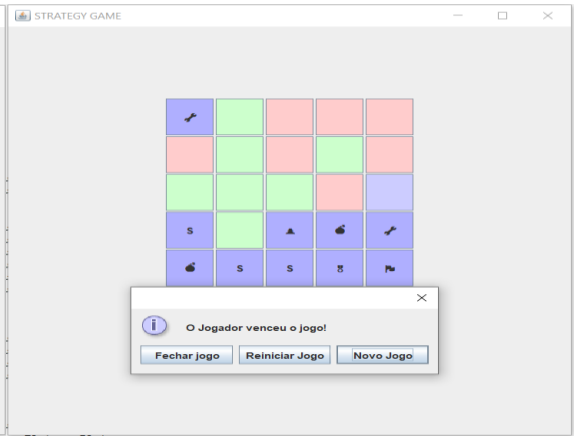
Tela com peças posicionadas (1) e Tela DEBUG (2)



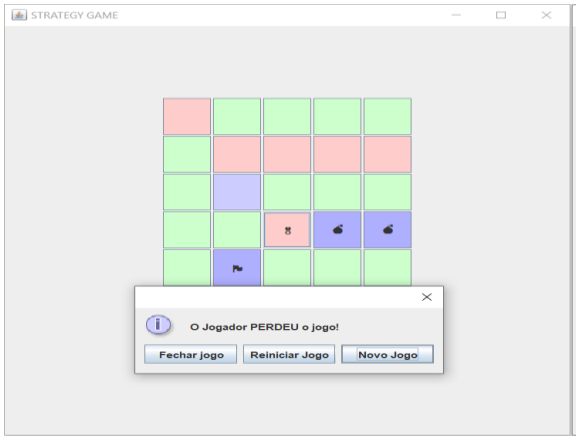


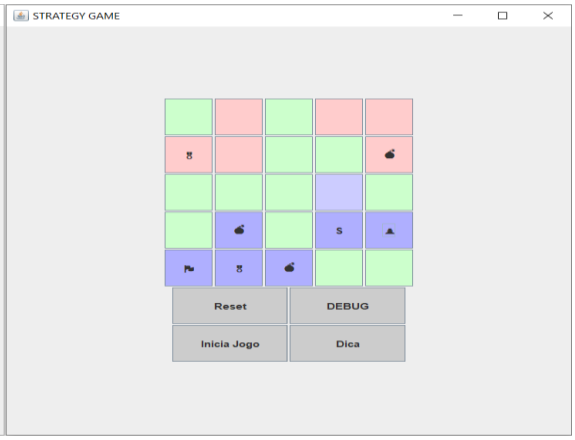
Tela Dica (1) e Tela Vitória (2)





Tela Derrota (1) e Tela Jogo (2)





Breve explicação em onde e como foram aplicados os conceitos de orientação objeto.

Um exemplo de polimorfismo que podemos mencionar é a classe CelulaFactory que como já foi explicado tem como função criar as células usadas durante todo o processo do jogo, então, dependendo do argumento passado para o método factory ele pode retornar diferentes tipos de objetos de células.

Outro conceito aplicado em nosso projeto, é a classe abstrata. No caso, foi criada uma classe abstrata chamada Peça que possui atributos sobre as peças usadas no jogo então ela pode ser somente usada por outras classes para que estas implementem seus métodos e use de seus atributos.

E por fim, o conceito de herança foi aplicado sobre as peças usadas no jogo (Soldado, Cabo, Bomba, etc.) essas classes recebem o acesso aos atributos e métodos da classe Peça.

Conclusão

No início do desenvolvimento do projeto nós tivemos dificuldades em conseguir estruturar o jogo de forma correta. Mas depois conseguimos contornar os problemas encontrados e nos organizarmos de forma a conseguir desenvolver o jogo em que cada um complementava o desenvolvimento do outro. Durante o desenvolvimento do projeto, aprendemos a importância de planejar bem o projeto, definir objetivos e trabalhar em equipe de forma que a dupla conseguisse alcançar os objetivos do projeto. Por fim, achamos que esse projeto foi muito importante para o desenvolvimento de nossas habilidades em programação orientada objetos e resolução de problemas. Achamos o desafio desse projeto interessante e uma boa agregação para a nossa formação como programadores.

Sugestão para Trabalho Futuro

Sugerimos o desenvolvimento e implementação de uma aplicação que simula um computador simples, onde sua interface gráfica simula um teclado e um monitor tal como de um notebook, todas as operações implementadas aparecem na “tela” do notebook e todos os inputs serão feitos pelo teclado desenvolvido.